Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Chess Puzzle Analysis (Interim report)

Author:
Mihhail Sorokin

Supervisor:
Dr. Nicolas Wu

Second Marker:
Benjamin Hou

April 9, 2024

Contents

Chapter 1

Introduction

Chess has long been analysed and enjoyed by both amateurs and professionals alike, and with the birth of computer science, an algorithm, along with a powerful enough computer to run it on, has been hunted for since at least the 1950s.[1] In 1997, Garry Kasparov, the strongest chess Grandmaster at the time, was defeated by Deep Blue,[2] and in the world of chess, humans were displaced by machines forever.

Despite this, chess remains a popular pastime for many people, with the largest online chess website, `chess.com`, having over 160,000,000 total members [3]. Whilst chess engines compete at the superhuman level with cutting-edge techniques, many average chess players refine their ability with study of previous positions – chess puzzles – where the goal is to find the critical move to punish the opponent's mistake and gain a decisive advantage, or win the game immediately via checkmate.

These puzzles seek to train the 'tactics' element of one's chess ability, and they can be categorised by the theme that occurs within them; some of these include *fork*ing enemy pieces with a knight, a well-known bishop sacrifice known as the *Greek gift sacrifice*, and many more, in addition to their combinations.[4] This allows players to identify which types of patterns they are often overlooking in real games, and to selectively train to detect those.

It is surprising that this categorisation is most often done manually, be it via community voting,[5] or a list of rules that the puzzle must satisfy to be classed one way or another.[6] This, along with the infamous saying that 'chess is 99% tactics', suggests that there is a need to explore this problem and provide in-depth categorisation of puzzles (beyond what is currently the norm).

Chess Grandmasters often talk about the 'feeling' of a position – is it possible to quantify this?

1.1 Challenges

(This section is blank... for now)

1.2 Contributions

(This one too)

Chapter 2

Background

The motivation behind solving chess puzzles, especially under time pressure, is to improve one's pattern recognition abilities. It has been shown [7] that chess players do not have a square-by-square, recollection of the chess board during play. Instead, they rely on interactions, and potential interactions, of pieces in a more abstract sense. This is made strikingly clear by a chess player's drawing of a position from memory. (Figure 2.1)

de Groot writes, "the pieces themselves do not appear in the drawing: rather the lines of force that go out from them". This heuristic allows expert human players to quickly hone in on the correct moves,[8] and significantly reduces their search space, when compared to a naive brute-force search, which is the most common strategy for chess engines. When analysing positions, even the most basic chess engines are able to calculate the most accurate line (except some pathological cases), but they are unable to draw similarities between different positions.

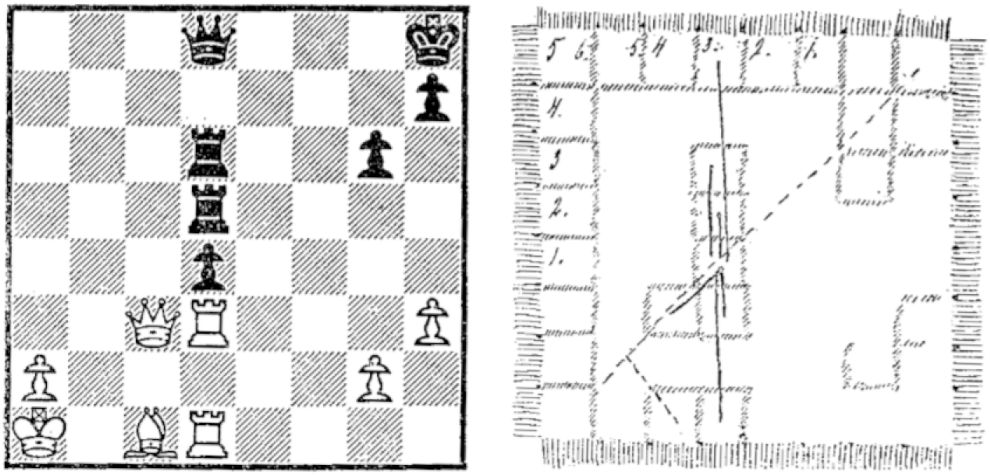

Figure 2.1: Taken from de Groot's 'Thought And Choice In Chess'.[7]

In Figures 2.2, 2.3, 2 chess positions are shown, featuring *back-rank checkmates* – one of the first tactical patterns that a beginner might learn. A castled king is forced to stay on the back-rank due to the configuration of his men, and an opposing heavy piece exploits this weakness by delivering checkmate. It is already non-trivial to programmatically identify this type of tactic, as a checkmate delivered to a king on the back rank is not necessarily a *back-rank checkmate*.

Given positions similar to ones like these, how does one draw similarities between the abstract relations of the heavy piece, king, and vulnerable back rank?

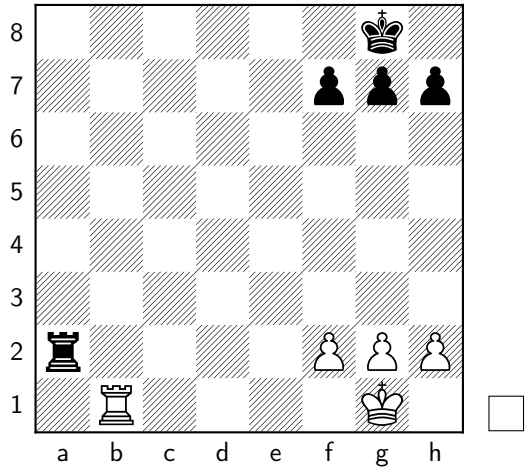

Figure 2.2: A trivial back-rank checkmate, white mates with **Rb8#** (TODO: Find real chess games where these positions/similar ones happened)

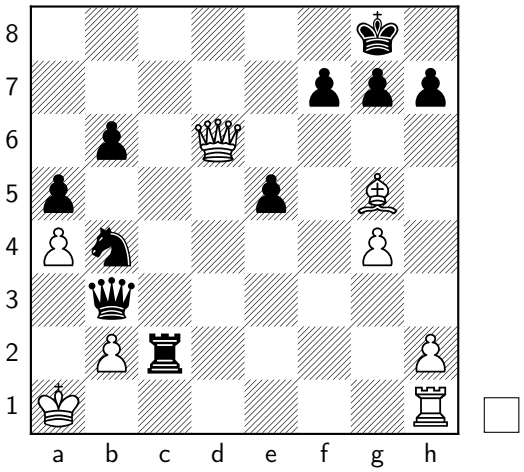

Figure 2.3: White mates with **Qd8#**.

2.1 Chess notation

TODO: Explain chess FEN, PGN notation. The reader is assumed to know how to play chess at a basic level

TODO: Show examples of basic puzzles/tactics in chess

TODO: Explain basic chessprogramming, e.g. bitboards, 0x88

2.2 A language for encoding piece relationships

In "A description language for chess",[9] López-Michelone et al. describe a language to search across chess positions. The main features of this language are descriptions for a chess piece attacking/defending another, attacking/defending a square, being located at a square, a square not being available for the enemy king, and the structure of white/black pawns on the board.

With their novel language, they are able to search a chess database for a pre-determined pattern, such as the *Greek gift sacrifice*, defined in the language as "kg8, pf7, pg7, B(ph7), Nf3, Qd1, Pe5"[1], which returns positions such as the one shown in Figure 2.4. After **16...Be7 17. O-O**, Black blundered with **17...Bxa3??**, after which, the *Greek gift sacrifice* (**18. Bxh7!!**, shown in Figure 2.5) was employed, eventually culminating in a win for White.

[1]This string corresponds to a black king on g8, black pawns on f7, g7, h7, a white bishop attacking h7, a white knight on f3 (ready to deliver a check with **Ng5**+, a common motif in *Greek gift sacrifices*), a white queen on d1, and finally, a white pawn on e5 to dislodge the usual black knight on f6.

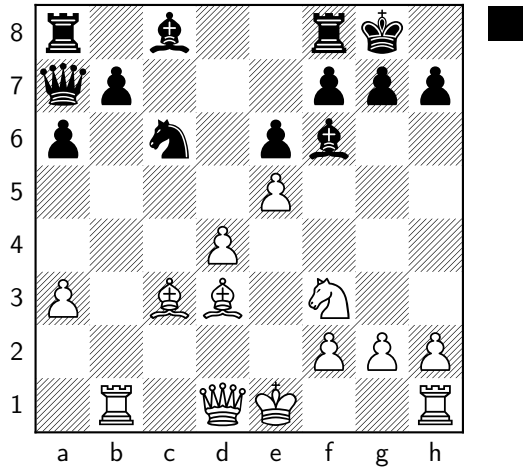

Figure 2.4: **Pirc, V – Porreca, G**, YUG-ITA m 1953, move 16.

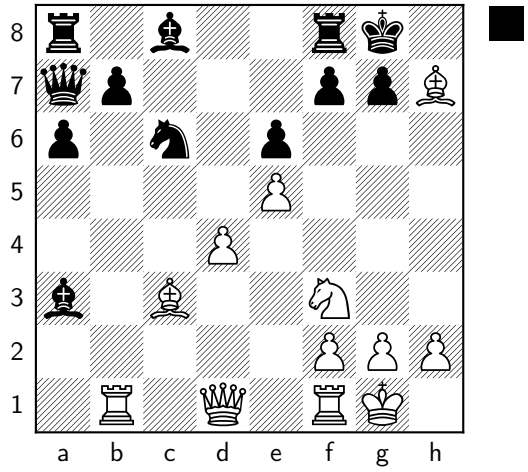

Figure 2.5: **Pirc, V – Porreca, G**, YUG-ITA m 1953, move 18. Black resigned after 6 moves.

Their language is also able to deal with some light variations, as it is able to identify the games shown in Figures 2.6, 2.7. In both of these positions, White has the brilliant move **1. Qh6+!!**, following with **2. Rh8#** if **1...Kxh6**, and either **2. Rf7#** or **2. Rb7+** (leading to a quick mate) if **1...gxh6**.

This pattern, whilst very rare, is undeniably identical between the 2 games. The unavailability of the **g6** square to the enemy king, combined with the harmony of White's pieces leads to the same tactic in both games.

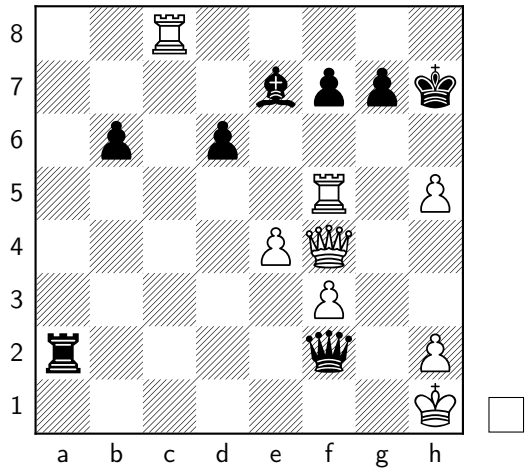

Figure 2.6: **Carlsen, M – Karjakin S**, World Chess Championship 2016, move 50.

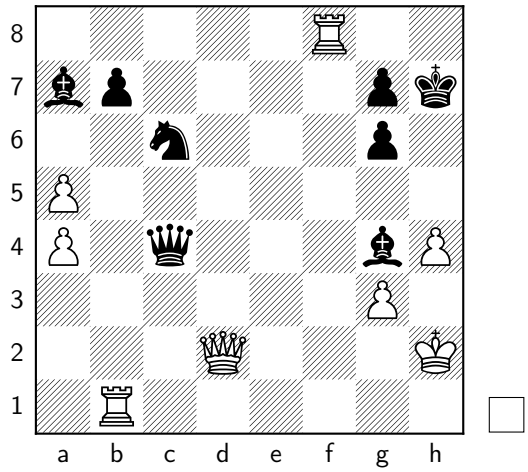

Figure 2.7: **Popov, N – Novopashin, A**, URS-ch otbor 1979, move 32.

The work of López-Michelone et al. is a promising proof of concept that shows the power of a language that allows to specify piece relationships on a more abstract level than previously possible. The biggest drawback of their solution, as mentioned by the authors, is the fact that this language still requires an expert with pre-existing extensive knowledge to encode the tactics into their language. The authors hypothesise that automatic recognition of these patterns is likely some sort of neural network, which is one of the many possible directions of this project.

2.3 CQL: Chess Query Language

The Chess Query Language (CQL), as invented by Costeff,[10] is another implementation of an advanced way to find chess positions in a given database. Since its inception in 2004, it has grown and is able to support very powerful, sometimes esoteric, queries to find predefined patterns.

TODO: Show examples from http://www.gadycosteff.com/cql/examples/smotheredmate.html and http://www.gadycosteff.com/cql/examples/turton.html

In addition to the Costeff's CQL, there is a from-scratch clone of CQL 6.1 which includes extra features (TODO: list some) and supports other chess variants.[11]

2.4 lichess.org puzzle database

https://lichess.org is a popular, open-source chess website which often publishes the games that have been played by players of all skill levels on it. As part of this, Lichess has published over 3.6 million rated and tagged puzzle positions.[5] To generate these, 300 million games were analysed with a powerful chess engine to find critical positions in which a move must be played to capitalise on the opponent's mistake. These puzzles were initially tagged to 124 manually created themes,[12] which were identified by a python implementation.[6]

As various users of the site solved the puzzles, and manually highlighted the themes that they felt occured in the puzzle, the ratings and tags of the puzzle database evolved until their current state.

This database is invaluable for this project, and can likely serve either as input, or as a baseline to compare the results to.

2.5 Recognition of the *Greek gift sacrifice* in chess games

Miroslav et al. report their findings on a program [13] to identify the *Greek gift sacrifice* in chess middlegames[2]. They found success, partly thanks to their detailed representatioon of the board, where each square is represented by 71 binary attribites [13] to encode the piece on the square and the possible squares which are reachable by this piece.

These attributes were also supplemented by 59 other binary attribites which were devised by expert chess players, and represent the more complex, but still quantifiable relationships.[13] These include open files, control of vulnerable squares, piece activity, and so on.

Miroslav et al. achieved an 87.7% classification performance on detecting whether a position features a successful *Greek gift sacrifice* on relatively small (<200 positions) datasets. This work is promising, but more investigation is needed given their analysis of only one pattern with a decision tree algorithm. Also, their program still relied on predetermined chess patterns, which introduces bias and might miss intricate similarities between positions.

2.6 A novel chess board representation for convolutional neural networks

In Sabatelli's thesis,[14] the effectiveness of neural networks to analyse whether a chess position is winning or losing is explored, without creating specific look-ahead algorithms. This is a very challenging task, but as part of the analysis, the author proposed manually encoded features to a convolutional neural network (CNN) to help identify strong patterns within the position.

Some of these include an extra feature if the opponent is in check, specifying the squares controlled by a piece exerting a pin on a different piece, centre control, and vulnerable squares.[14] These are well known heuristics that are often taught to beginner-intermediate chess players, and the author claims that these additional layers are "extremely representative of the chess positions", and cause the CNN to outperform a naive, fully-connected neural network.

[2] A hard-to-quantify phase of the chess game where most pieces are developed and the kings are positioned away from danger.

This work is also a promising result, as it shows that these chess patterns, albeit manually quantified, are valuable for an algorithm when analysing a given chess position.

2.7 Chess moves as kernels for texture classifier

In this study, Turker et al. propose novel kernels for efficient feature extraction in the task of texture detection.[15] These 5x5 kernels are directly based on the move a rook, bishop, knight, and their combinations.

Whilst not directly applicable to the context of chess puzzle analysis, this work shows that it may be possible to include these kernels in an CNN-based analysis of the chess position. It would be interesting to apply to the other CNN chess work (e.g. Sabatelli's thesis,[14] discussed above) and analyse its effect on the success of the technique.

2.8 TODO: Research + summarise more papers

2.9 Ethical discussion

This project has few, if any, ethical considerations that need to be made. The biggest concern is the source of the large datasets that may be used in this project, namely, the Lichess puzzle database.[5] However, this database, along with all Lichess games is released under the Creative Commons CC0 license – meaning there is no restriction in its use.

Each puzzle in this database contains a link to the game from which it was extracted, and this contains a Lichess username, which is deanonymising. In the initial data processing stage, these links are going to be removed, as they are unnecessary for the goals of the project.

Another concern is the use of, and reference to specific chess games and the players in them, and various chess compositions[3] with their authors. Other than proper reference to the creators of the game or composition, there are no other considerations to be made.

[3] An artificially created position to showcase a uniquely challenging, rare, or otherwise interesting theme.

Chapter 3

Planning

Note: this section will be should be removed after the interim report!

3.1 Project plan

3.1.1 Research more literature

There are still a number of papers that attempt to come up with new ways to algorithmically analyse chess positions and these should also be reviewed. There is also a very large amount of papers to do with psychology of professional chess players, and the biggest conclusion of these, the fact that chess players often analyse the piece patterns and links, not the squares [7], has been the backbone of many papers. It's possible that there are still untapped findings that could be explored by this project.

3.1.2 Decide on deterministic/graph-based vs CNN approach (or both?)

So far, I have come across 2 main ways in which previous work has dealt with problems similar to this one. The first is a way to logically encode piece relationships [4], [8], [9], [10], [6], which can prove to be difficult without manually encoding patterns. The second is a machine learning approach, usually with CNNs, since they tend to effectively 'identify patterns' [14], [15], [13]. Since my project concerts itself only with a stationary position, without accounting for lookforward moves, it's possible that some sort of encoding can be use (in fact, there is a paper called Chess2Vec [16] which has attempted to predict chess moves – not analyse positions).

3.1.3 Implement a prototype to categorise puzzles by lichess label and difficulty, using their puzzle tagger as a helper

A well written prototype should be created to implement the algorithm described. The implementation language depends on the algorithm chosen, but my initial guess is that any logical solution would lend itself to Haskell (or another FP language), whilst the most natural language for any machine learning approach is python (almost definitely with the pytorch library, as I have the most experience with it).

A soft requirement for a language for this is to have a well-made chess library to analyse moves, pieces, etc. Implementing one from scratch is hopefully not too difficult, but reinventing the wheel is unnecessary.

3.1.4 Given a set of positions, attempt to cluster them into labels. The labels are either manual or automatic(?) or even absent(? unsupervised?)

This is part of the implementation and is in essence, using the output of the written program to analyse a set of chess problems.

3.1.5 Experiment with the other approach, contrast and compare

Time permitting, the other approach (either graph-based or CNN) could be analysed. The outputs, advantages, disadvantages, etc. of the 2 methods should be compared and contrasted here – it's possible that one of them works much better than the other.

3.2 Evaluation plan

3.2.1 Quantitative

The simplest metric to check if the solution is correct would be to compare against lichess's 3.6million database. It is possible there exist other chess position databases (this should be investigated) to compare to. For obvious reasons, this will be split into training/test (if the algorithm requires training on positions). Lichess's puzzle tags can be treated as a multi-label classification algorithm, and as long as the project produces a similar output, this can be used. The labels and difficulty can also be used to create a basic notion of distance based on lichess's classification, which should then be compared with the project's results.

Some papers researched in the background section (and possibly ones that are yet to be read) have attempted a similar problem with chess position analysis. The results of this project can be then compared against their solution, hopefully on the same benchmark.

3.2.2 Qualitative

Multiple positions can be designed/retrieved from a database and analysed by a chess player. Of course, the player's skill dictates the complexity of positions which they are able to sufficiently analyse and compare, but with the help of modern chess engines, it should be possible for an intermediate player who is aware of tactic and puzzle classifications to draw conclusions. These could then be compared to the predictions of the program implemented by the project and conclusions can then be drawn.

Bibliography

[1] Claude E Shannon. XXII. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, 1950.

[2] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

[3] Chess.com. Members - chess.com. https://www.chess.com/members, 2024. [Online; accessed 24-January-2024].

[4] Manuel Cristóbal López-Michelone and Jorge Luis Ortega-Arjona. Patterns for the game of chess. In *Proceedings of the 11th Latin-American Conference on Pattern Languages of Programming*, pages 1–15, 2016.

[5] lichess.org. lichess.org open puzzle database. https://database.lichess.org/#puzzles, 2022. [Online; accessed 24-January-2024].

[6] lichess.org. lichess.org puzzle tagger. https://github.com/ornicar/lichess-puzzler/blob/master/tagger/cook.py, 2023. [Online; accessed 24-January-2024].

[7] Adriaan D De Groot. *Thought and choice in chess*, volume 4. Walter de Gruyter GmbH & Co KG, 1965.

[8] Merim Bilalić, Robert Langner, Michael Erb, and Wolfgang Grodd. Mechanisms and neural basis of object and pattern recognition: a study with chess experts. *Journal of Experimental Psychology: General*, 139(4):728, 2010.

[9] Manuel Cristóbal López-Michelone and Jorge L Ortega-Arjona. A description language for chess. *ICGA Journal*, 42(1):2–13, 2020.

[10] Gady Costeff. The chess query language: CQL. *ICGA Journal*, 27(4):217–225, 2004.

[11] Gamble, Robert. CQLi. https://cql64.com/, 2022. [Online; accessed 24-January-2024].

[12] lichess.org. lichess.org puzzle tag description. https://github.com/lichess-org/lila/blob/master/translation/source/puzzleTheme.xml, 2021. [Online; accessed 24-January-2024].

[13] Miroslav Kubat and Jan Žižka. Learning middle-game patterns in chess: A case study. In *Intelligent Problem Solving. Methodologies and Approaches: 13th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2000 New Orleans, Louisiana, USA, June 19–22, 2000 Proceedings 13*, pages 426–433. Springer, 2000.

[14] Matthia Sabatelli. *Learning to play chess with minimal lookahead and deep value neural networks*. PhD thesis, Faculty of Science and Engineering, 2017.

[15] Turker Tuncer, Sengul Dogan, and Volkan Ataman. A novel and accurate chess pattern for automated texture classification. *Physica A: Statistical Mechanics and its Applications*, 536: 122584, 2019.

[16] Berk Kapicioglu, Ramiz Iqbal, Tarik Koc, Louis Nicolas Andre, and Katharina Sophia Volz. Chess2vec: learning vector representations for chess. *arXiv preprint arXiv:2011.01014*, 2020.